Block 2

FH University of Applied Sciences TECHNIKUM WIEN

Supervised Machine Learning-pipeline

illustrated by kNN (& linear regression)

Lecturer: --
Authors: Lackner Stefan, Bernhard Knapp
Credits: David Meyer, Pascal Plank

AI

Supervised learning

Regression
KNN regression
Regression trees
Linear regression
Multiple regression
Ridge and Lasso regression
Neural networks

Classification
KNN classification
Classification trees
Ensembles & Boosting
Random Forest
Logistic regression
Naive Bayes
Support vector machines
Neural networks

Non-supervised learning

Clustering
k-means
Hierachical clustering
DB-scan

Dimensionality reduction
PCA / SVD
tSNE
Multi dimensional scaling
Linear discriminant analysis

Machine learning process
Data handling
EDA, data cleaning
Training and testing
Feature selection
Class balancing
etc

Generative AI
Not covered here

Reinforcement learning
Not covered here

FH University of Applied Sciences TECHNIKUM WIEN

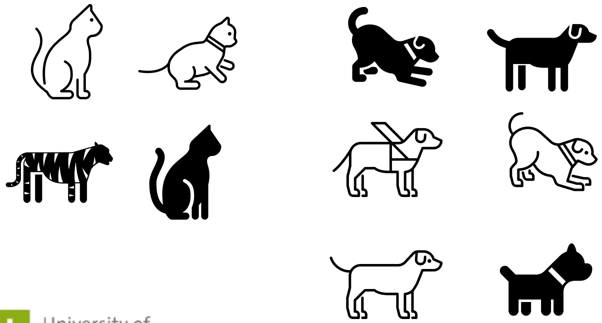
Machine Learning Types

- **Supervised Learning**:
  - labelled data
  - Direct feedback
  - Predict an outcome/future, forecasting
  - E. g. predict customers that will return

labelled training data

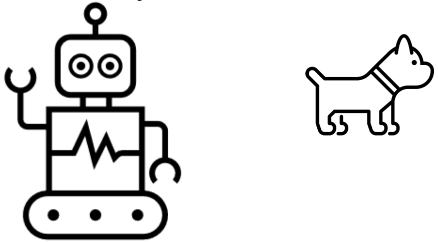
cats

dogs



I think that this is a dog.

(based on the training data that I have seen before)



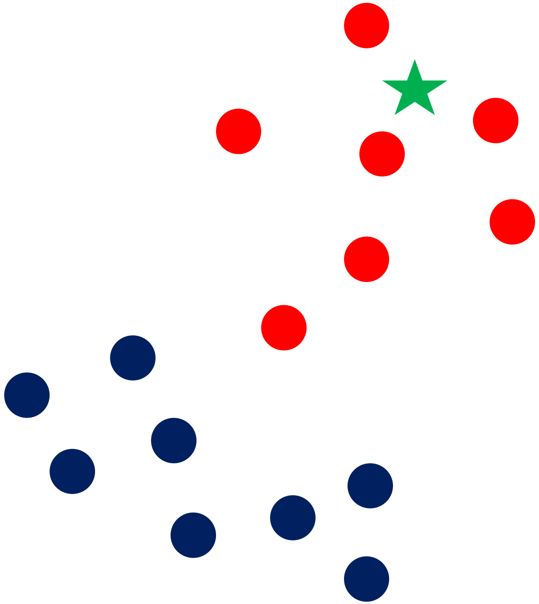
Model

Feedback: correct!

FH University of Applied Sciences TECHNIKUM WIEN

First some k-nearest neighbors (kNN) intuition:

Once upon a time there was a king with a rose garden …

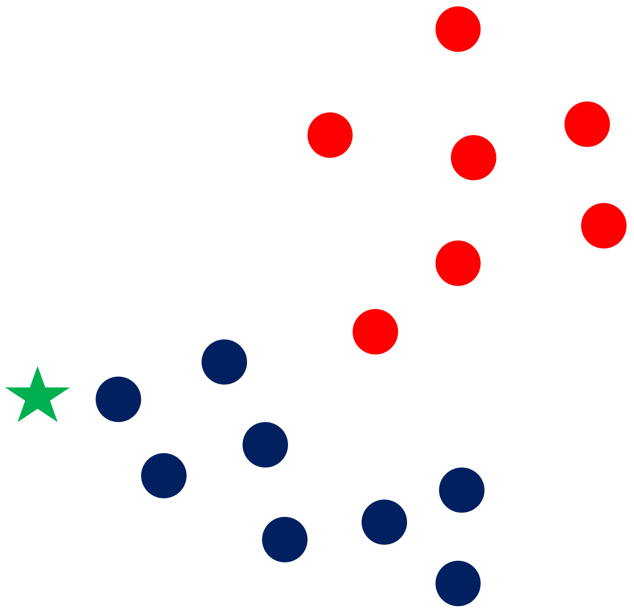


“That seed will clearly become a red rose!“

Red rose

Blue rose

Rose seed of unknown colour

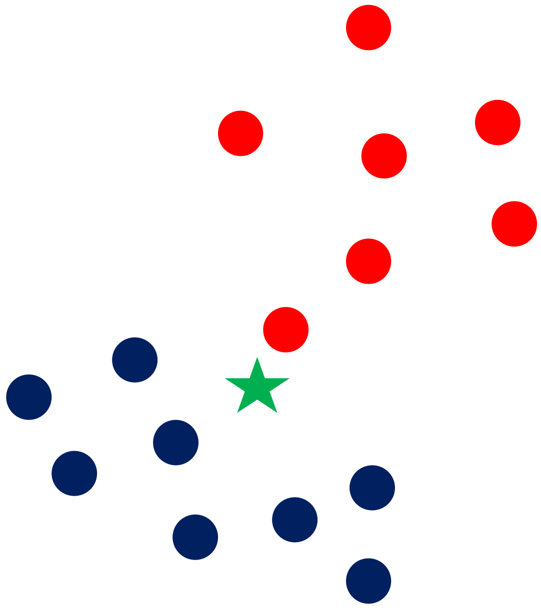


“That seed will clearly become a blue rose!“

Red rose

Blue rose

Rose seed of unknown colour



“Hmmm …..“

Determine the colour of the seed based on the k nearest neighbours (=kNN) …
(but in reality we have many more than 2 dimensions therefore it is not that obvious to a human)

… still many people consider kNN as one of the simplest AI algorithms.

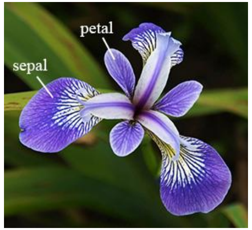
Red rose
Blue rose
Rose seed of unknown colour

But before we dive deeper into kNN we have to introduce some terminology …

Contents

FH University of Applied Sciences TECHNIKUM WIEN

Features & Targets



- In supervised machine learning the data consists of:
- **Features** (aka. predictors), usually denoted by ***X***
- **Targets** (regression), **labels** (classification), usually denoted ***y***

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
54	37	15	2	setosa
69	31	54	21	virginica
45	23	13	3	setosa
64	32	45	15	versicolor
71	30	59	21	virginica
50	23	33	10	versicolor
65	32	51	20	virginica
55	26	44	12	versicolor
67	33	57	25	virginica
44	32	13	2	setosa

target/label, ***y***

predictors/features, ***X***

FH University of Applied Sciences TECHNIKUM WIEN

Features & Targets

- Targets/lables have to be collected which is often expensive
- The general aim is to **learn a function** mapping from features/pedictors to targets/labels
- The learned function is then **applied** on data where targets/lables are **unknown** and returns the most likely target value/label
- The learned function is also refered to as the **fitted model**. It is simply the algorithm you chose with parameters adjusted to the training set

FH University of Applied Sciences TECHNIKUM WIEN

Features & Targets

#1 Training the algorithm

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
54	37	15	2	setosa
69	31	54	21	virginica
45	23	13	3	setosa
64	32	45	15	versicolor
71	30	59	21	virginica
50	23	33	10	versicolor
65	32	51	20	virginica
55	26	44	12	versicolor
67	33	57	25	virginica
44	32	13	2	setosa

#2 Use the fitted algorithm for predcition

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
79	38	64	20	
77	38	67	22	
77	28	67	20	
77	30	61	23	?
77	26	69	23	
76	30	66	21	
74	28	61	19	
73	29	63	18	

$$\hat{Y} = \hat{f}(X)$$

Unfortunately, the definitions in the literature defer …

X

features

feature variables

independent variables

predictor variables

regressors

explanatory variables

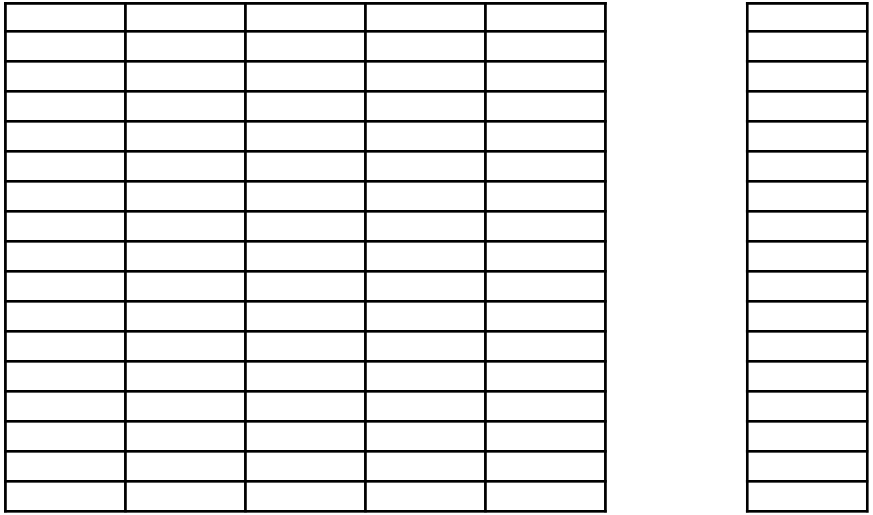
input variables

exogenous variables

y

target variable
target
dependet variable
predicted variable
regressand
explained variable
output
response variable
label
endogenous variable

samples
or
rows



Python‘s numpy (e.g. in sklearn.linear_model.LinearRegression)
X : numpy array or sparse matrix of shape [n_samples,n_features]
y : numpy array of shape [n_samples, n_targets]

This exists/works also for regressions …

Regression example:

X
features

y
target variable

Samples

	Age	Education level	Years experience	Manager of	Sick days
John Smith	25	2	1	0	3
Kate Mayer	37	5	15	5	0
Angelo Biden	32	0	2	0	21
…	…	…	…	…	…

Income / y
39 356
77 834
25 899
…

University of Applied Sciences TECHNIKUM WIEN

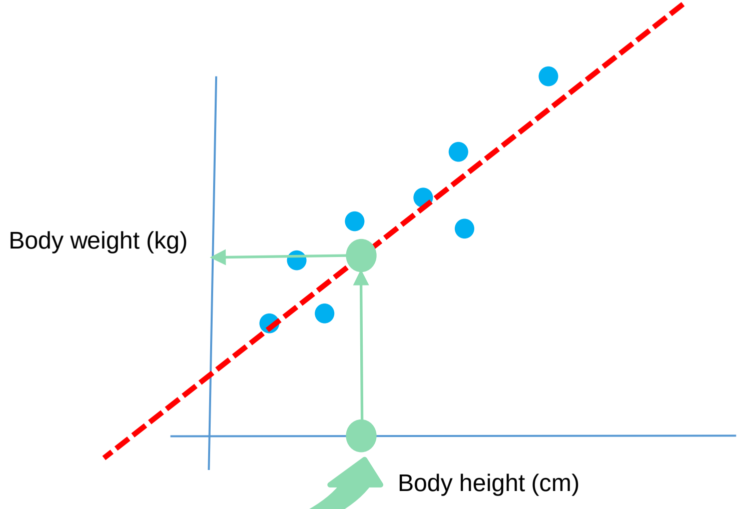
Python‘s numpy (e.g. in sklearn.linear_model.LinearRegression)
X : numpy array or sparse matrix of shape [n_samples,n_features]
y : numpy array of shape [n_samples, n_targets]

Regression vs classification?

- If we try to predict a **number** (e.g. 43.71 or 99) we talk about **regression**
- If we try to predict a **class** (e.g. car / cycle / boat) we talk about **classification**

Simple Linear Regression: idea

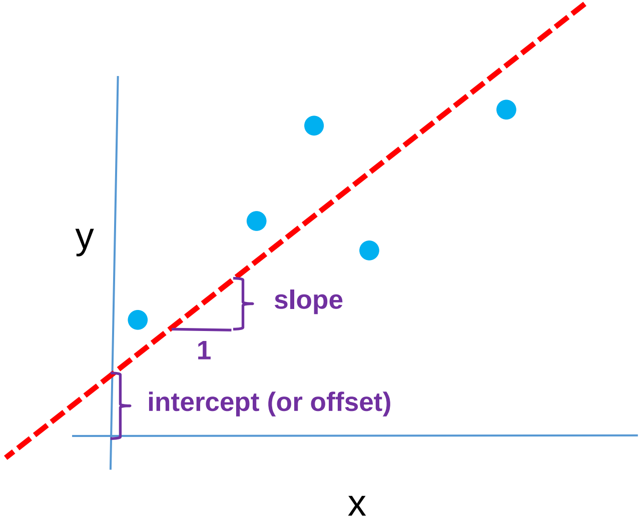
person id	Body height (cm)	Body weight (kg)
1	191	86
2	173	75
3	165	63
4	177	76
5	181	80
6	152	48
7	182	72
…		
236	169	???



Estimate “body weight“ based on “body height“

FH University of Applied Sciences TECHNIKUM WIEN

A line is defined by two parameters:



$$y = slope*x + intercept$$

or

$y = kx + d$

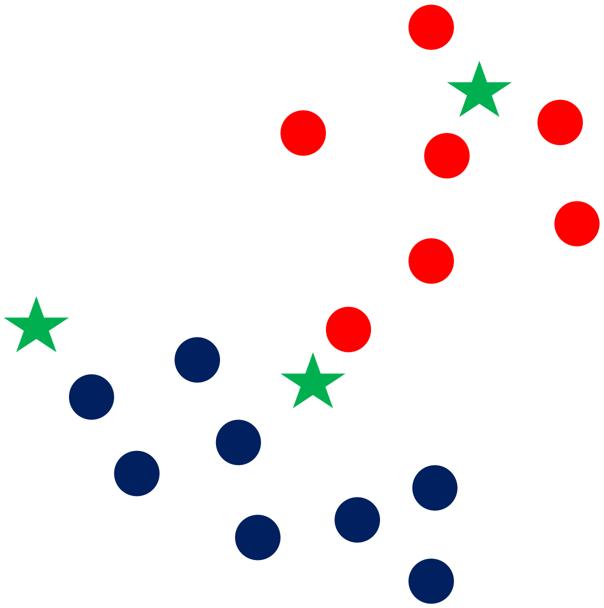
$y = ax + b$

$y = b_0 + b_1x$

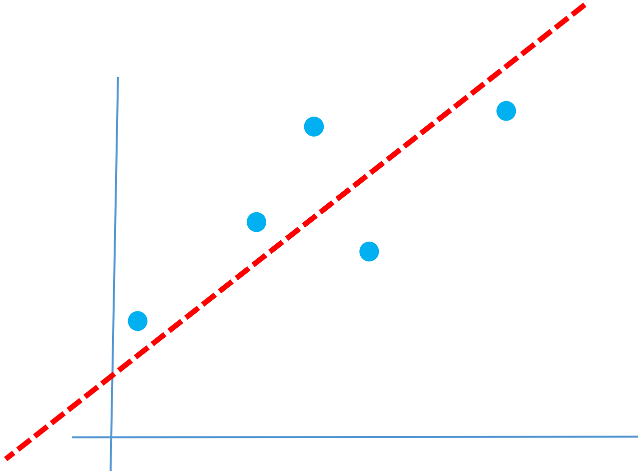
Slope and *intercept* are called parameters.

(In later lectures we will try to find optimal values for them to fit our data)

So what would be our *features* and *targets* in the rose example above?
And is it regression or classification?



So what would be our *features* and *targets* in the income example above?
And is it regression or classification?

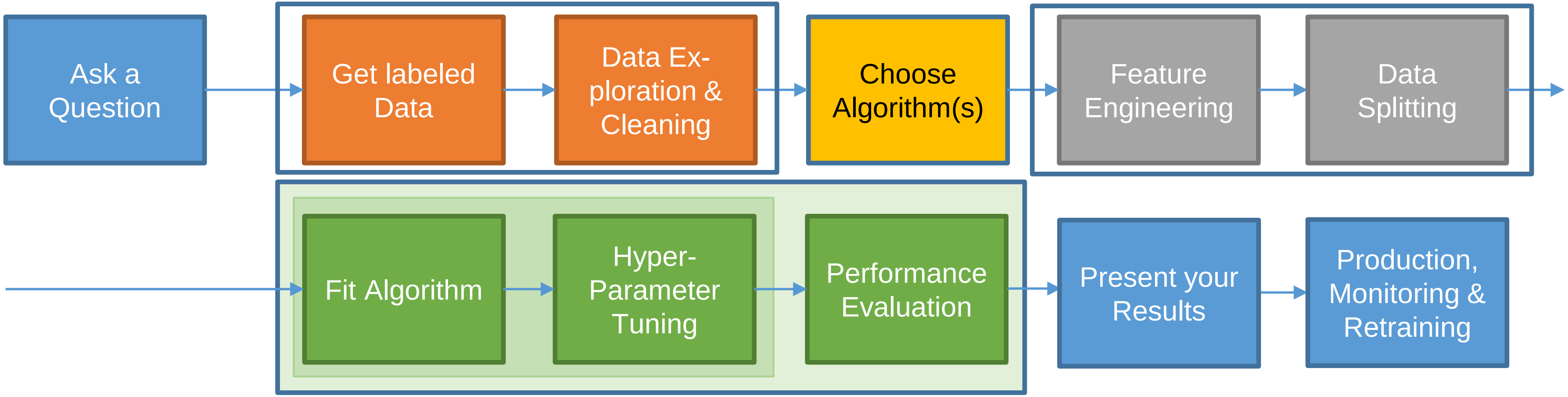


FH University of Applied Sciences TECHNIKUM WIEN

Contents

FH University of Applied Sciences TECHNIKUM WIEN

Supervised Machine Learning Workflow



In reality the process will not be linear, you‘ll jump back and forth between the steps

FH University of Applied Sciences TECHNIKUM WIEN

On the following slides thinking for each step how this would work for our roses (classification) and income (regression) example!

Supervised Machine Learning Workflow

Get labeled Data

- If the data you have for training leaves out important cases you have a problem!
- Your data must **truthfully represent the whole problem** you want to solve!
- No algorithm can solve this problem for you!

Supervised Machine Learning Workflow

Choose Algorithm(s)

- Choosing an algorithm introduces a decision into the workflow. This decision is called the **inductive bias**.
- No algorithm is always superior. Search for the „**No free lunch theorem**“ by Wolpert.
- Often, you‘ll simply have to **try more than one algorithm**

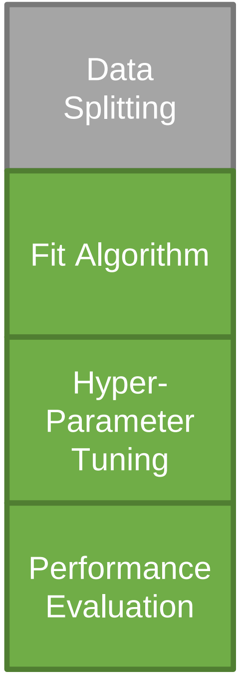
Supervised Machine Learning Workflow

Choose Algorithm(s)

- Algorithms have **parameters** and **hyper-parameters**
- **Parameters** are estimated from the data
- **Hyper-parameters** must be set by the user. Optimization can only be done by **trying several different settings**!

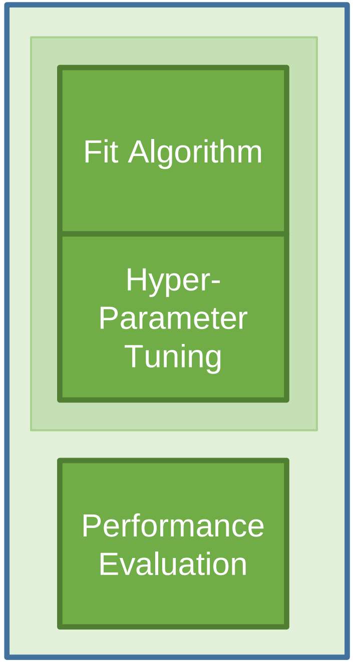
FH University of Applied Sciences TECHNIKUM WIEN

Supervised Machine Learning Workflow



- **Data Splitting:** Data needs to be partitioned into (1) **training**, (2) **validation** and (3) **test sets**
- **Fit Algorithm:** Training sets are used to estimate/optimize paramters from the data
- **Hyper-Parameter Tuning**: Validation sets are used for **performance evaluation** of several **specific hyper parameter settings** and selection of the best hyper parameter setting
- **Performance Evaluation**: Test sets are used to assess the performance of the algorithm with the best found hyper-parameter setting. **We need to choose a performance metric**!

Supervised Machine Learning Workflow



- **Training** and **tuning** should be **separated conceptually**
- In **practical workflows** they are executed as **one step**
- For algorithm tuning one also needs to **measure performance to** choose the best hyper-parameter setting. This is **technically the same** as performance assessment
- Performance evaluation is **conceptually different**, since it is done **after the best hyper-parameter was chosen** using a different data partition

Supervised Machine Learning Workflow

Get labeled Data

Does your data represent the true population?

Choose Algorithm(s)

Inductive bias,
no free lunch,
which hyper-parameter values should be tried?

Data Splitting

Several options exist! (more later)

Performance Evaluation

Which performance metric to use?
Are all errors equal?

FH University of Applied Sciences TECHNIKUM WIEN

Contents

FH University of Applied Sciences TECHNIKUM WIEN

Demo 1

SciKit Learn

- load_iris() … Load and return the iris dataset (classification).
- yourAlgorithm = KNeighborsClassifier() (or sklearn.linear_model.LinearRegression())
- yourAlgorithm.fit() … fits the Algorithm to the data
- yourAlgorithm.predict() … predicts labels/target values to feature values

Try to get this running in Python!

FH University of Applied Sciences TECHNIKUM WIEN

Block 3

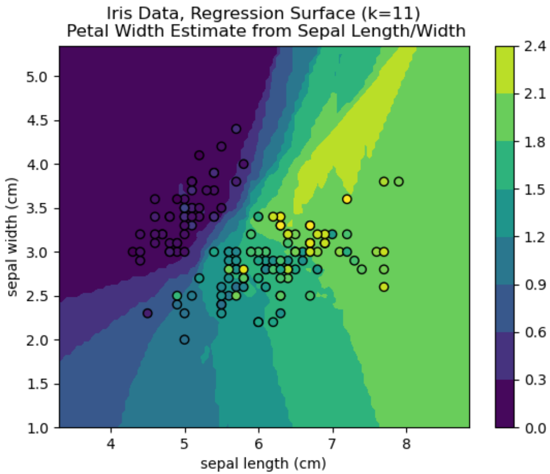
FH University of Applied Sciences TECHNIKUM WIEN

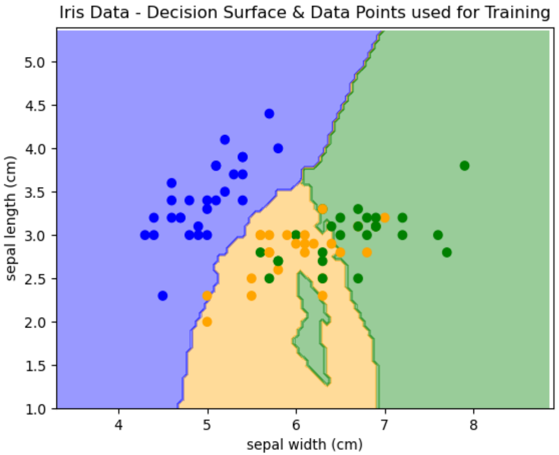
Contents

FH University of Applied Sciences TECHNIKUM WIEN

Performance Metrics

- Measure **how well our algorithm performs** on a given dataset
- Return **a scalar value** as a performance summary
- **Fundamentally different** for **regression** (predicting a continuous variable e.g. income) and **classification** (predicting categorical variable e.g. colour of flower)





Performance Metrics - Regression

Mean Absolute Error (MAE)

This could be used for our linear regression example about income but not for the roses!

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i|$$

divide by the number of e.g. persons

sum up over all e.g. persons

take the absolute value

e.g. [True income of person i in our test set] - [Predicted income of person i of our algorithm]

FH University of Applied Sciences TECHNIKUM WIEN

Performance Metrics - Regression

Mean Squared Error (MSE) or Mean Squared Deviation (MSD)

$$\text{MSD} = \quad \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Using the square makes our metric more sensitive to outliers

Root Mean Square Deviation (RMSD) or Root Mean Square Error (RMSE)

$$\text{RMSD} = \quad RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

RMSE is just sqrt(MSE)

FH University of Applied Sciences TECHNIKUM WIEN

Performance Metrics - Regression

More Metrics are available

Regression metrics

See the Regression metrics section of the user guide for further details.

metrics.explained_variance_score(y_true, ...)	Explained variance regression score function.
metrics.max_error(y_true, y_pred)	max_error metric calculates the maximum residual error.
metrics.mean_absolute_error(y_true, y_pred, *)	Mean absolute error regression loss.
metrics.mean_squared_error(y_true, y_pred, *)	Mean squared error regression loss.
metrics.mean_squared_log_error(y_true, y_pred, *)	Mean squared logarithmic error regression loss.
metrics.median_absolute_error(y_true, y_pred, *)	Median absolute error regression loss.
metrics.mean_absolute_percentage_error(...)	Mean absolute percentage error regression loss.
metrics.r2_score(y_true, y_pred, *[, ...])	R^2 (coefficient of determination) regression score function.
metrics.mean_poisson_deviance(y_true, y_pred, *)	Mean Poisson deviance regression loss.
metrics.mean_gamma_deviance(y_true, y_pred, *)	Mean Gamma deviance regression loss.
metrics.mean_tweedie_deviance(y_true, y_pred, *)	Mean Tweedie deviance regression loss.

FH University of Applied Sciences TECHNIKUM WIEN

API Reference — scikit-learn 0.24.2 documentation

Performance Metrics - Classification

This could be used for our roses example but not for the income example!

Confusion Matrices

- Are the starting point for deriving performance measures in classification
- An example would be Covid-19 test (or a red rose!):

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

Performance Metrics - Classification

Confusion Matrices

From these 4 outcomes lots of performance metrics can be deduced:

		Predicted condition		Sources: [13][14][15][16][17][18][19][20] view · talk · edit	
	Total population $= P + N$	Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2 PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI) $= \frac{TP}{TP + FN + FP}$

Confusion matrix - Wikipedia

… confused now? Try: https://www.youtube.com/watch?v=Kdsp6soqA7o&t=1s

Performance Metrics - Classification

Accuracy

- Simplest measure
- Sensitive to class balancing

ACC = (TP+TN) / (TP+TN+FP+FN)

… this is what most people intuitively come up with: The proportion we got correct divided by everything

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	**True positive (TP)**, hit	**False negative (FN)**, type II error, miss, underestimation
	Negative (N)	**False positive (FP)**, type I error, false alarm, overestimation	**True negative (TN)**, correct rejection

FH University of Applied Sciences TECHNIKUM WIEN

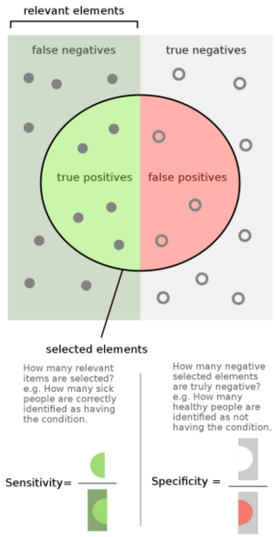
Performance Metrics - Classification

True Positive Rate (TPR, sensitivity) / True Negative Rate (TNR, specificity)

- Interesting if one event is more impotant than the other. E.g.
  - in **spam filtering** it is more important that (almost) all real emails land in the inbox and a few false negatives (spam in your inbox) do not matter much (TNR is important)
  - but at an **airport** it is important to catch every terrorist and you rather check thousand passengers again manually (even though they are not terrorists) instead of missing one terrorist (TPR is important).

- There is a trade-off between TPR/TNR. This is reflected in the **ROC** curve.

[Confusion matrix - Wikipedia]



TPR=TP/P TNR=TN/N

TPR=TP/TP+FN TNR=TN/TN+FP

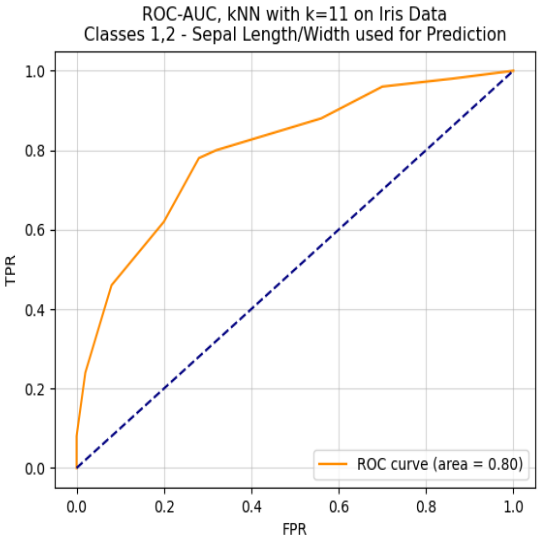
FH University of Applied Sciences TECHNIKUM WIEN

[More metrics here: API Reference — scikit-learn 0.24.2 documentation]

Performance Metrics - Classification

- **Reciever Operator Charateristc (ROC) curves** relate **TPR** and **FPR** in one plot. Left upper corner is the sweet spot
- Used for binary classification problems

TPR



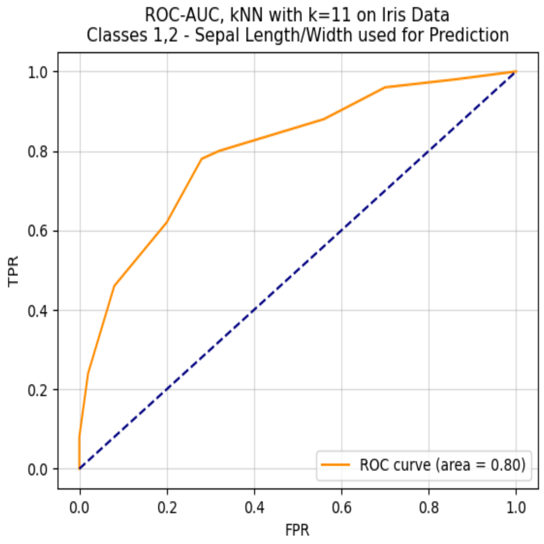
1 – TNR (=FPR)

[Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) | Teachers College Columbia University]

FH University of Applied Sciences TECHNIKUM WIEN

Performance Metrics - Classification

- The **area under the ROC (AROC) curve** is a popular measure for comparing predictor performance:
  - Perfect performance: AROC = 1
  - “Good“: AROC > 0.8
  - Random prediction: AROC = 0.5
  - Problem in the data if AROC < 0.5 (e.g. flip in signs)



[Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) | Teachers College Columbia University]

FH University of Applied Sciences TECHNIKUM WIEN

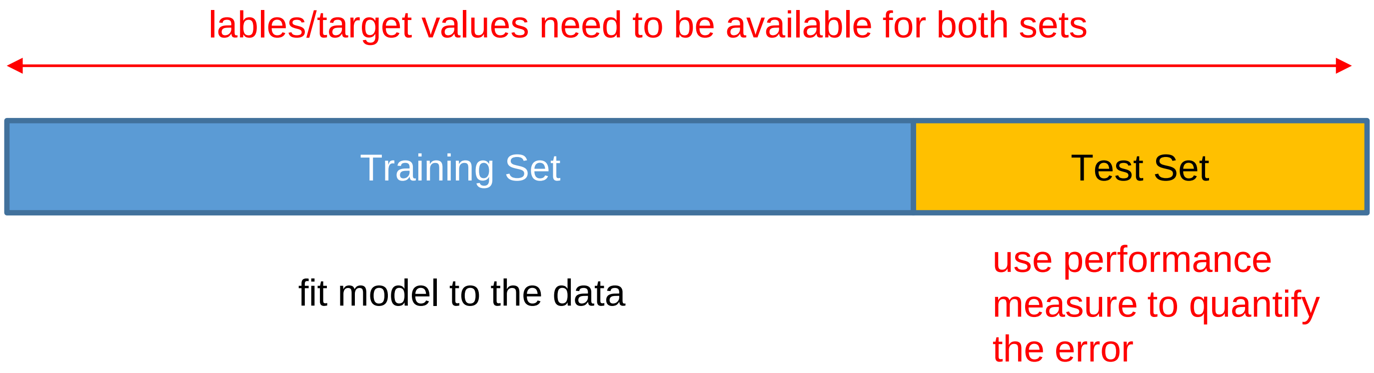
Contents

FH University of Applied Sciences TECHNIKUM WIEN

Performance Evaluation - train/test split

- Our predictions won‘t be perfect!
- We need to quantify how good our fitted model/function/algorithm will be on **unseen** data
- To do so, we need to introduce a **training phase** and a **testing phase** together with a **train/test split of the data (Hold-Out Validation)**

E.g. we train on the income of Peter, John and Jane and we test how well we predict Peters (known) income.



Performance Evaluation

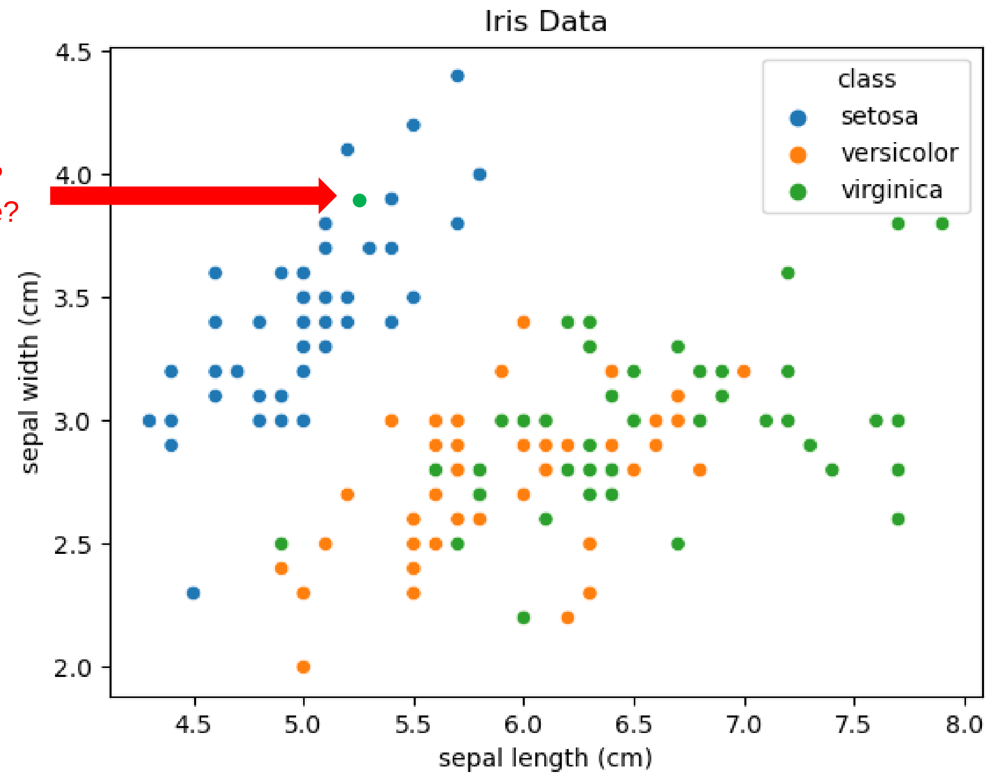
- Performance can also be measured on the same data on which the model was trained (=**resubstitution evaluation**)
- However, this estimate will be **overly optimistic**!
- Results from resubstitution evaluation are called **training error/accuracy** or **in-sample error/accuracy**
- Even if resubstitution evaluation results are good, our algorithm might have only learned the specifics of the training data including noise.

  - Resubstitution evaluation is bad
  - Always split between training and test data

FH University of Applied Sciences TECHNIKUM WIEN

Performance Evaluation

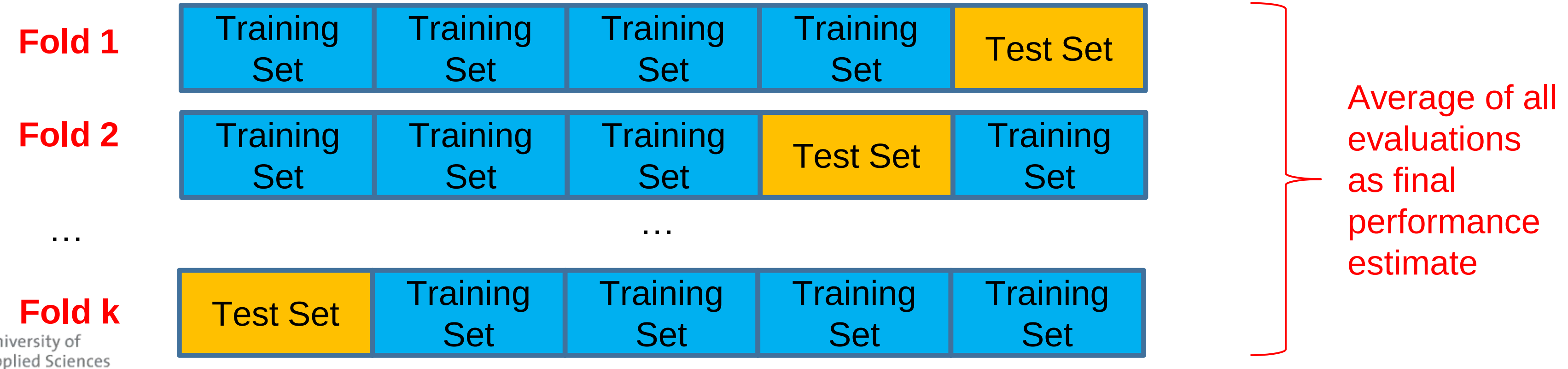
Might this be a data error?
Should we be perfect here?



FH University of Applied Sciences TECHNIKUM WIEN

Performance Evaluation

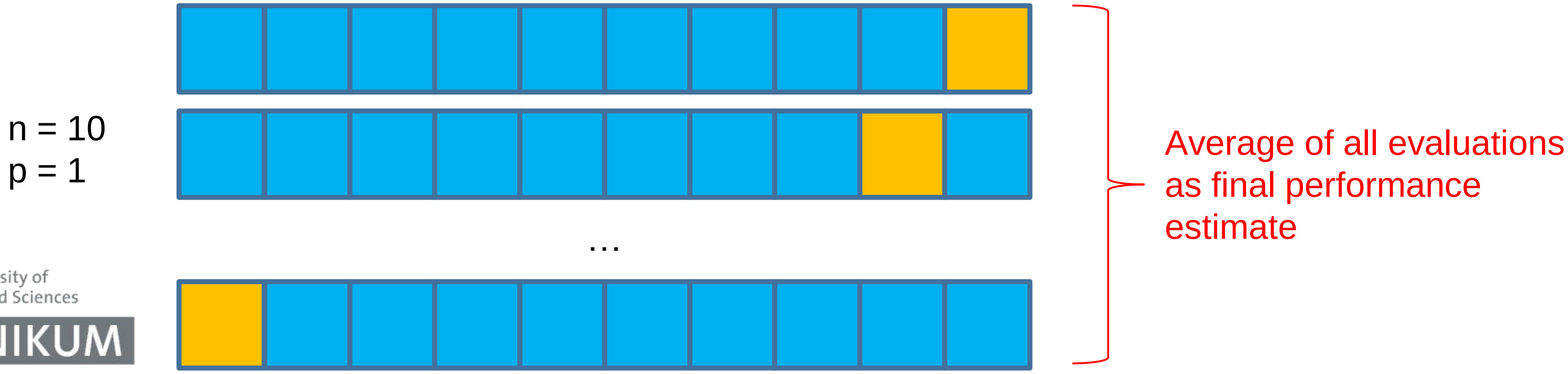
- Using a simple train/test split (hold-out validation) is not robust since the split is arbitrary
- To get a more robust estimate we need to repeat the train/test split
- One way to do this is **k-fold cross-validation (KfCV)**



FH University of Applied Sciences TECHNIKUM WIEN

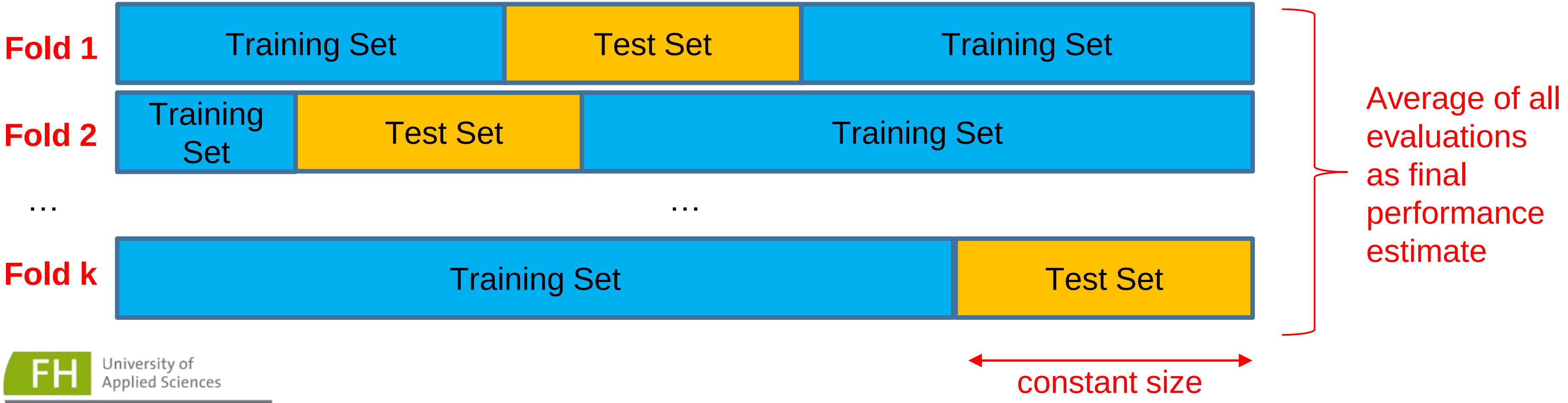
Performance Evaluation

- Turning k-fold cross-validation on its head we can also talk about **leave p-out cross-validation (LpOCV)**
- E.g., the number of folds could be equal to the number of data points. This is called **leave-one-out cross-valildation (LoOCV)**
- When using e.g. 2 data points for performance evaluation, this would be L2OCV



Performance Evaluation

- Folds can also be **sampled randomly** without a fixed startification
- This is called **Monte Carlo Cross-Validation (MCCV)**
- Size of test set is constant



Performance Evaluation - Recap

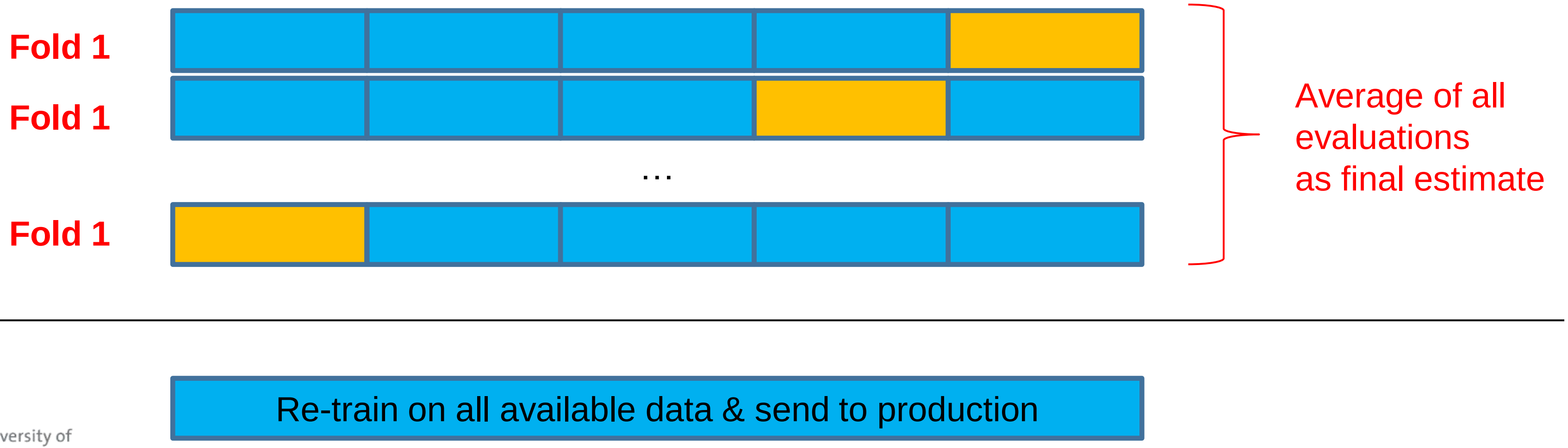
- **Resubstitution evaluation → ** forbidden
- **Holdout Validation (=1-Fold Cross-Validation) →** unstable estimates
- **Leave 1-out Cross-Validation →** used regularily, single estimates not independent due to data overlap, computationally burdensome.
- **K-Fold Cross-Validation →** used regularily with k=5, k=10. Higher k not nescessarily better since single evaluation in folds become more and more dependent due to data overlap.
- **Monte Carlo Cross-Validation →** used regulary, robust results

FH University of Applied Sciences TECHNIKUM WIEN

Performance Evaluation

Obtaining one final model after KfCV, LpOCV, MCCV

- Usually the algorithm is retrained on the whole data set before it is put to production!



Contents

FH University of Applied Sciences TECHNIKUM WIEN

Demo 2

Demo 1 and …

- train_test_split() … partition your data set
- sklearn.model_selection.KFold()
- LeaveOneOut()
- LeavePOut()
- sklearn.metrics.accuracy_score(y_true, y_pred)
- sklearn.metrics.classification_report()
- etc

FH University of Applied Sciences TECHNIKUM WIEN

Block 4

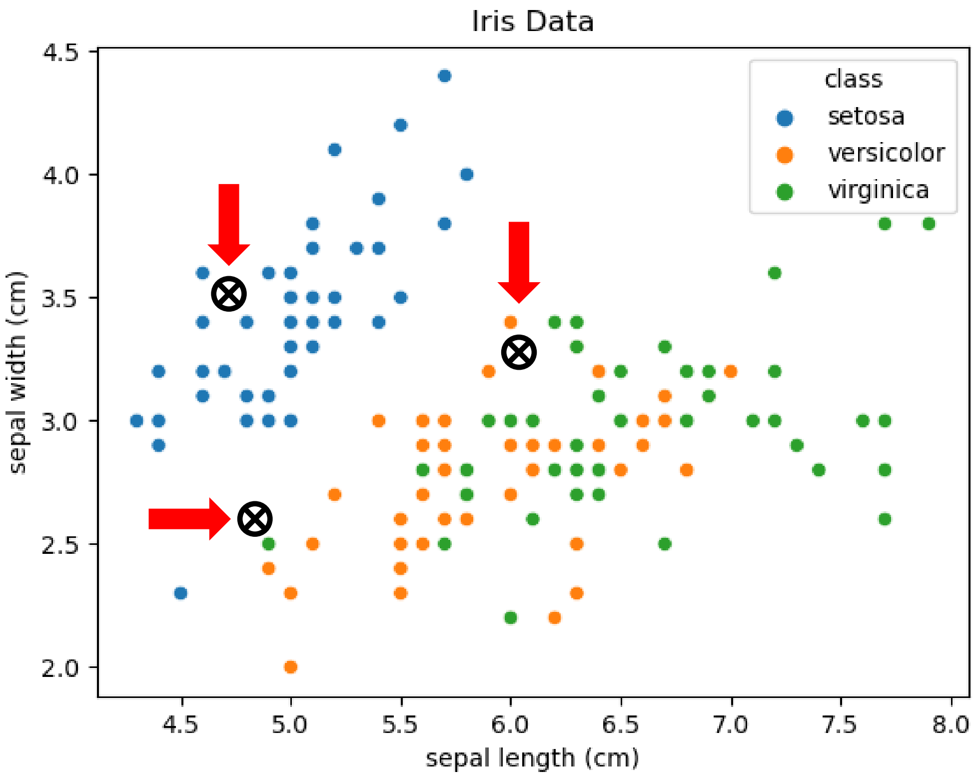
Contents

FH University of Applied Sciences TECHNIKUM WIEN

kNN for Classification (Basic Idea)

- kNN = k-Nearest Neighbors Algorithm
- kNN is based on the notion of **similarity**, where similarity is usually expressed as the **inverse of a chosen distance metric** (for now, just think about the euclidean distance)
- A datapoint with unknown label is classified by **comparing it with ist k-nearest neighbors** with known labels
- k is a user defined **hyper**-parameter

kNN for Classification (Basic Idea)



Classify the 3 points visually using k = 1,3,5
What is important about choosing k?

kNN for Classification (Basic Idea)

Algorithm:

1. Load data (with known features and targets (=labels))

2. Split data into a training set and a test set

3. Choose a value for k

4. For each point in the test set:
   a) find the Euclidean distance to all training data points
   b) store the Euclidean distances in a list and sort it
   c) choose the first k points
   d) assign a class to the test point based on the majority of classes present in the chosen points

5. Compare the predicted labels of the tests set with the true labels of the test set

FH University of Applied Sciences TECHNIKUM WIEN

kNN - Distance and Similarity

- kNN is based on similarity/distance
- Similarity is understood as the inverse of distance and vice versa
- From a mathematical perspective we need a s.c. **distance metric**.
- Distance metrics must satisfy several **conditions**

$$D(a, b) \geq 0$$

$$D(a, b) = 0 \text{ iff } a = b \ldots \text{identity}$$

$$D(a, b) = D(b, a) \ldots \text{symmetry}$$

$$D(a, b) + D(b, c) \geq D(a, c) \ldots \text{triangle inequality}$$

FH University of Applied Sciences TECHNIKUM WIEN

kNN - Distance and Similarity

Euclidean Distance (L-2 Norm, real valued data)

$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

$$D(a,b) = \sqrt{\sum_{i=1}^{p}(a_i - b_i)^2} \ldots \text{p} = \text{number of dimensions}$$

Manhatten Distance (L-1 Norm, real valued data)

$$D(a,b) = |a_1 - b_1| + |a_2 - b_2|$$

$$D(a,b) = \sum_{i=1}^{p} |a_i - b_i| \ \ldots \ \text{p} = \text{number of dimensions}$$

FH University of Applied Sciences TECHNIKUM WIEN

kNN - Distance and Similarity

Minkowsky Distance(s) (real valued data)

$$D(a,b) = \sqrt[k]{\sum_{i=1}^{p} |a_i - b_i|^k} \text{ ... p = number of dimensions}$$

Cosine Similarity (real valued data)

$$csim(a,b) = \frac{a \odot b}{\|a\| \|b\|} \text{ ... } \odot \text{ dot product , } \| a \| \text{ norm of a}$$

$$csim(a,b) = \frac{\sum_{i=1}^{p} a_i b_i}{\sqrt{\sum_{i=1}^{p} a_i^2} \sqrt{\sum_{i=1}^{p} b_i^2}} \text{ ... } a_i, b_i \text{ usually } \geq 0 \text{ and } \vec{a}, \vec{b} \neq \vec{0}$$

FH University of Applied Sciences TECHNIKUM WIEN

kNN - Distance and Similarity

Kullback Leibler Divergence (distributions)

$$D(P \parallel Q) = KL(P, Q) = \sum_{x \in X} P(x) \cdot \log \frac{P(x)}{Q(x)}$$

$x \in X$ ... concrete values of X (e.g. bin ranges from a histogram)

P, Q ... discrete distributions over X (e.g. probability of bins in a histogram)

kNN - Distance and Similarity

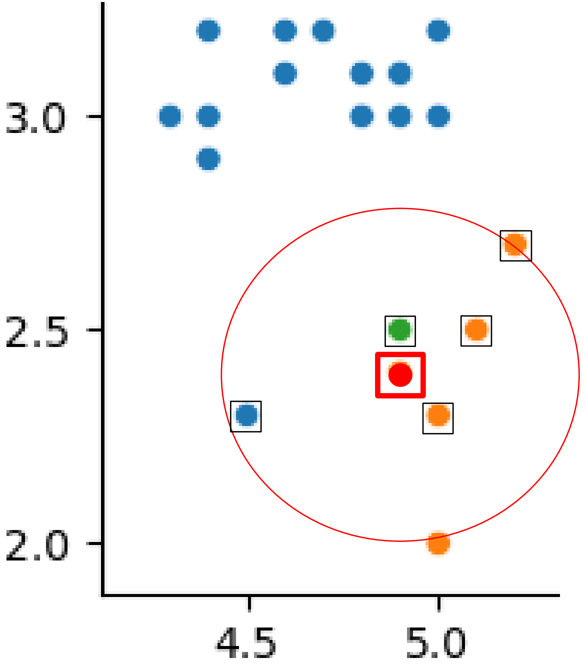
Very high dimensional spaces are problematic

- **Distance calculation** becomes burdensome (use approximate kNN, use dimensionality reduction)
- **Distances contract** (get more and more similar) when dimensionality grows very large.
- **High dimensional geometry is very unituitive**

Contents

FH University of Applied Sciences TECHNIKUM WIEN

kNN - Weighting



- Some neighbors are much closer to x‘ than others
- Closer neighbors could (should?) be more important

kNN - Weighting

Inverse distance weighting for classification (IDW)

$$w_i(x) = \frac{1}{d(x,x_i)}$$ … Do you see any problem here?*

$$w_i(x) = 1 \text{ if } d(x, x_i) = 0$$

$$P(y = c|x) = \frac{1}{k} \sum_{i \in N} w_i \cdot I(y_i = c)$$

c … a particular class, e.g. "cat"

k … number of neighbors

N … the neighborhood of point x

I … indicator Function

FH University of Applied Sciences TECHNIKUM WIEN

[* k nearest neighbour - How does scikit-learn's kNN model handle zero-distances when using inverse distance weighting? - Cross Validated (stackexchange.com)]

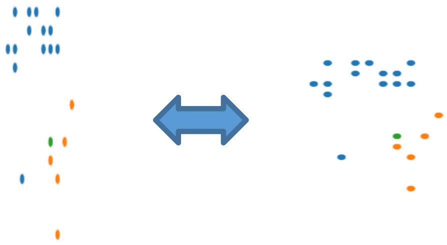
Contents

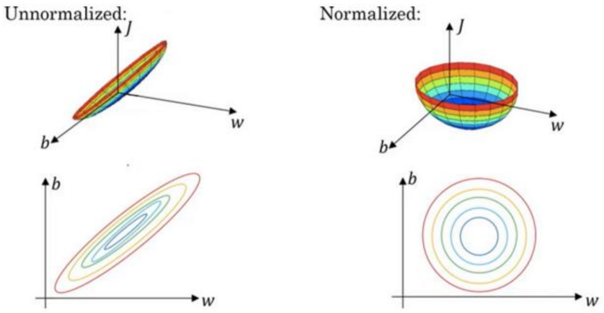
FH University of Applied Sciences TECHNIKUM WIEN

Feature Engineering

Feature Scaling (= normalisation)

- Feature scaling transforms data values s.t. they lie in the same range
- kNN is sensitive to attribute scaling since attributes with higher values are more influential when calculating distances.
- But: scaling does not always lead to better performance!
- Scaling is also relevant for other algorithms, e.g. In Neural Nets convergence can be accelerated by scaling





source: deeplearning.ai | Andrew Ng

kNN – Feature Engineering

Feature Scaling

- Standardization

$$x' = \frac{x-\mu}{\sigma} \;\ldots \mu = \text{mean of x}, \sigma = \text{standard deviation of x}$$

- Mean Normalization

$$x' = \frac{x-\mu}{max(x)-min(x)} \;\ldots\; \mu = \text{mean}$$

- Min-Max Normalization

$$x' = \frac{x-min(x)}{max(x)-min(x)}$$

FH University of Applied Sciences TECHNIKUM WIEN

What happens to the data after each type of scaling?

Contents

FH University of Applied Sciences TECHNIKUM WIEN

Hyper-Parameter Tuning

- **Recap**: Hyper-Parameters are set by the user and are **NOT** estimated from the data while fitting the model
- **Examples of Hyper-Paramters**: k in kNN, maximum depth in Decision Trees, learning rate in NNs, activation function in NNs, NN-Architecture
- **Examples of Parameters** (which are directly estimated from the data): Coefficients in linear models, split values and attributes in Decision Trees, weights in NNs
- **Hyper-Parameter Tuning is a form of optimization so we need to prevent overfitting**! For performance evaluation we need to use a 3-way split of the data in (1) a **training set**, (2) a **validation set** and (3) a **test set**

FH University of Applied Sciences TECHNIKUM WIEN

Hyper-Parameter Tuning

Option 0, Holdout Analog

#0 All available, labeled data

Split into train test

#1 Training Set | Test Set

Split into train, validation

#2 Training Set | Validation Set | Test Set

For all chosen settings of hyper-parameters: Train model and evaluate performance on the validation set. **The test set is untouched! → Select the best HP-setting**

#3 Training Set | Test Set

For the best HP-setting found in step #2: Retrain the model using the wohle Training set and evaluate performance on the test set.

Hyper-Parameter Tuning

Option 0, Holdout Analog

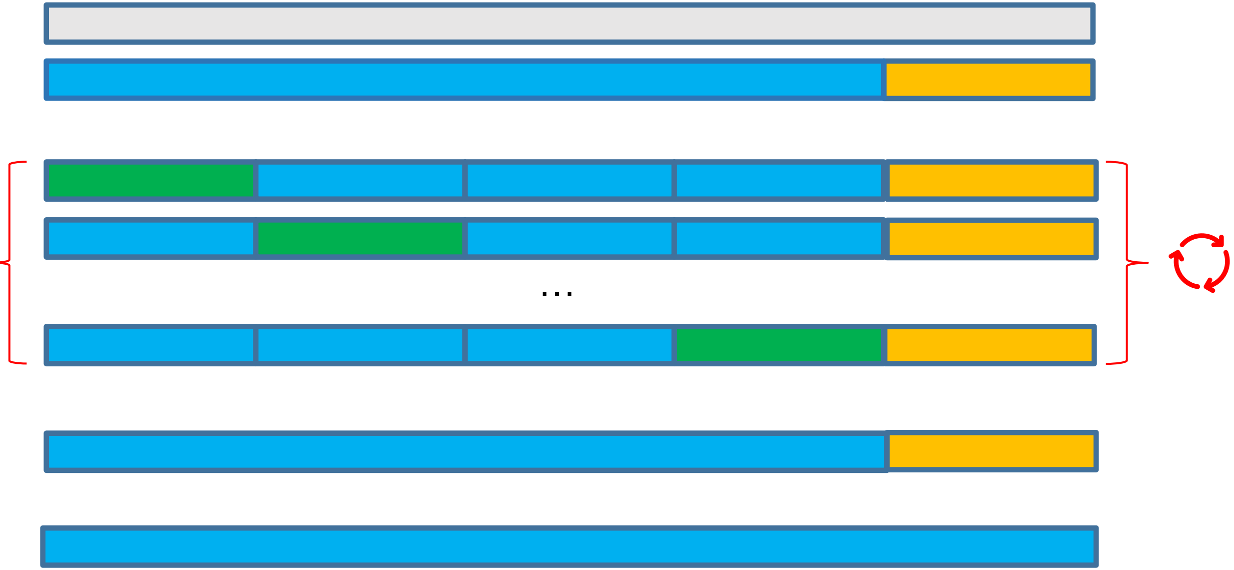
#4 Training set

For the best HP-setting found in step #2: Retrain the model using ALL the data.
Report the accuracy found in step #3 as model accurracy

- This approach is similar to holdout validation
- Both for HP selection and for performance assessment is has the same problems: **Poor robustness since splits are arbitrary**

Hyper-Parameter Tuning

Option 1, improve stability for HP-selection via Cross-Validation



#0

#1

#2, Cross-Validation

For all chosen settings of hyper-parameters: Train and evaluate performance via cross-validation. **The test set is untouched! → Select the best HP-setting**

#3

Perform holdout validation

#4

Retrain on the whole data, Report performance from step #3

Hyper-Parameter Tuning

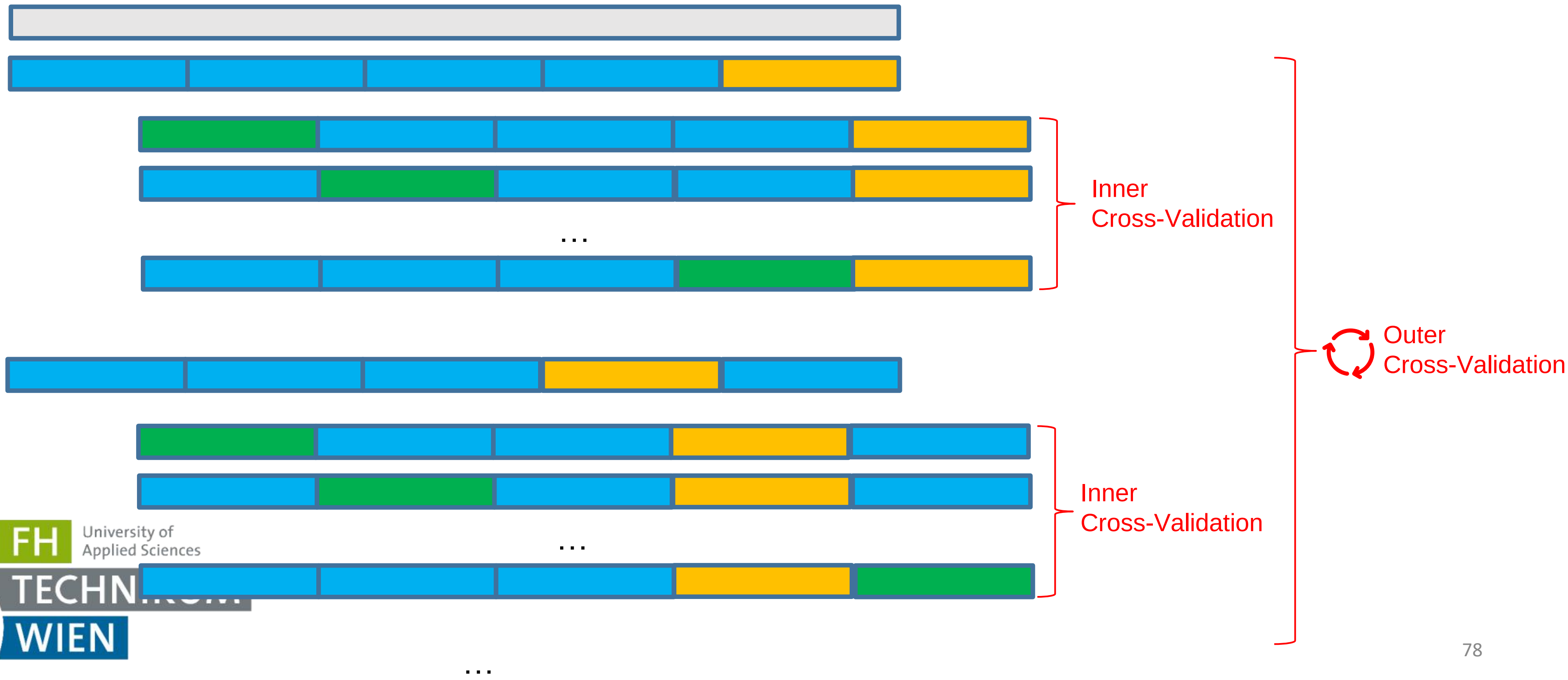
Option 1, improve stability for HP-selection via Cross-Validation

- In fact a combination of holdout validation (outer) and cross-validation (inner)
- Problem: performance evaluation is still based on one split
- Computational burden increases but is often ok for small/medium data sets. If we have **2 HPs with 3 settings** each and we choose **5-fold CV**, we need to fit **45 + 2 models**

FH University of Applied Sciences TECHNIKUM WIEN

Hyper-Parameter Tuning

Option 2, nested cross validation



Hyper-Parameter Tuning

Option 2, nested cross validation

- Both HP-selection and performance evaluation are robust due to cross-validation
- Computationally burdensome: If we have **2 HPs with 3 settings each** and choose **5-fold cross-vaidation** for inner and outer loops, we need to fit 3*3*5*5 = **225 + 1 models**

FH University of Applied Sciences TECHNIKUM WIEN

Hyper-Parameter Tuning

Grid-Search vs. Random-Search

- **Grid search** refers to the exhaustive combination of all possible HP settings. Quickly becomes infeasable.
- **Random search** refers to random sampling from all possible HP-setting combinations. Is often used as a alternative to grid search when many HPs must be tuned or models are slowly trained.
- **Heuristics** (e.g. GA)

FH University of Applied Sciences TECHNIKUM WIEN

Hyper-Parameter Tuning

kNN and hyper-parameters

- k is often referred to as the only HP in kNN
- However, choosing a distance metric is also a HP!
- The same is true for using weighted or unweighted aggregation!
- Even for a simple algorithm like kNN tuning everything can become impractical, especially for larger data sets!

FH University of Applied Sciences TECHNIKUM WIEN

Questions:

- Is kNN classification or regression?
- Is linear regression classification or regression?

- Which performance metrics can you use for kNN?
- Which performance metrics can you use for linear regression ?

- Is cross validation needed for kNN?
- Is feature engineering needed for kNN?

- Is cross validation needed for linear regression ?
- Is feature engineering needed for linear regression ?

- Does kNN have parameters?
- Does kNN have hyper-parameters?

- Does linear regression have parameters?
- Does linear regression have hyper-parameters?

FH University of Applied Sciences TECHNIKUM WIEN

How would a kNN-regressor work?

You have the necessary knowledge by now – think by yourself!

Contents

FH University of Applied Sciences TECHNIKUM WIEN

Demo 3

- kNN has several HPs: k, metric and weighting
  class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)¶
- Try StandardScaler(), MinMaxScaler() for data preparation
- GridSearchCV(), RandomSearchCV() can be used with cross_validation_score() to perform nested cross validation

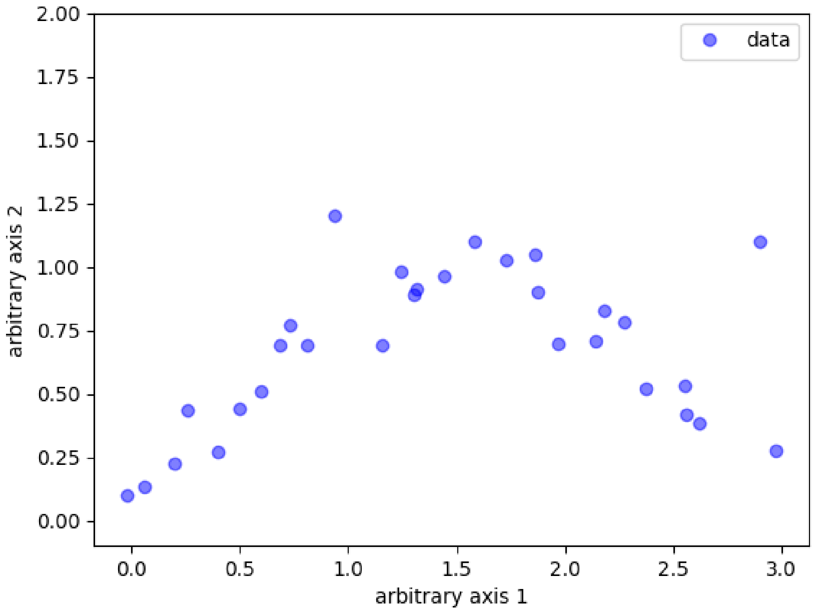
FH University of Applied Sciences TECHNIKUM WIEN

Contents

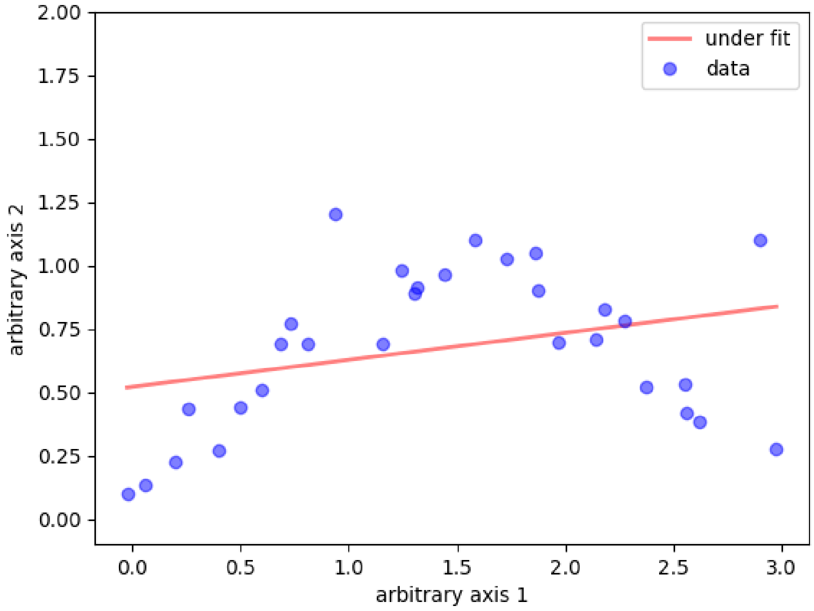
FH University of Applied Sciences TECHNIKUM WIEN

Overfitting

Which curve describes these points best?

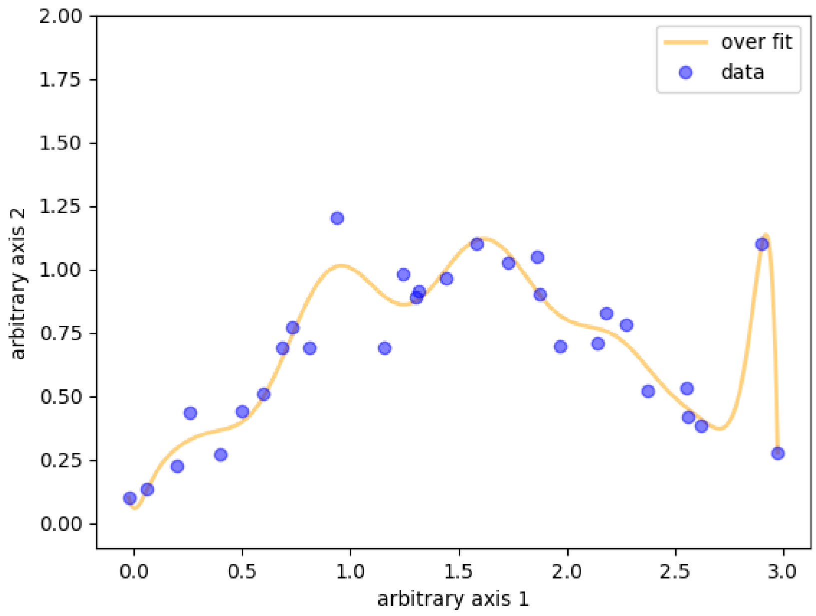


Possibly a too simple model (=underfitting)?

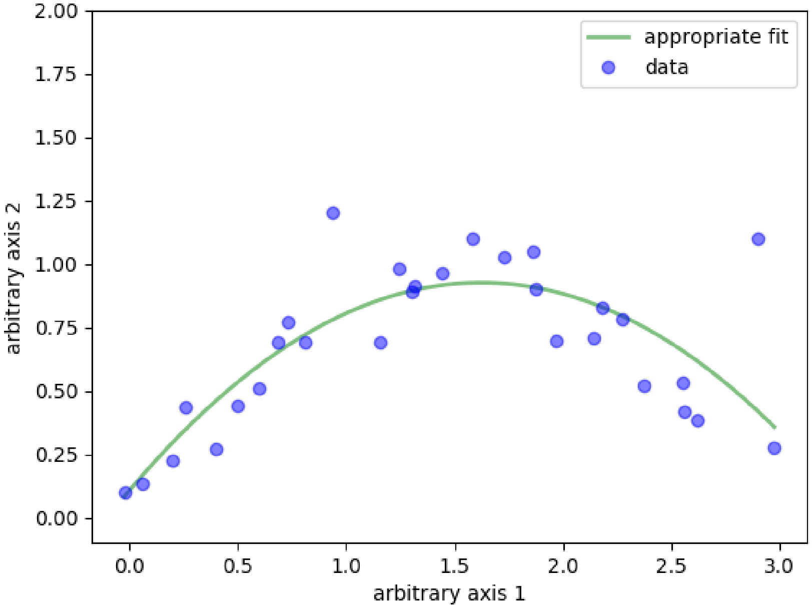


FH University of Applied Sciences TECHNIKUM WIEN

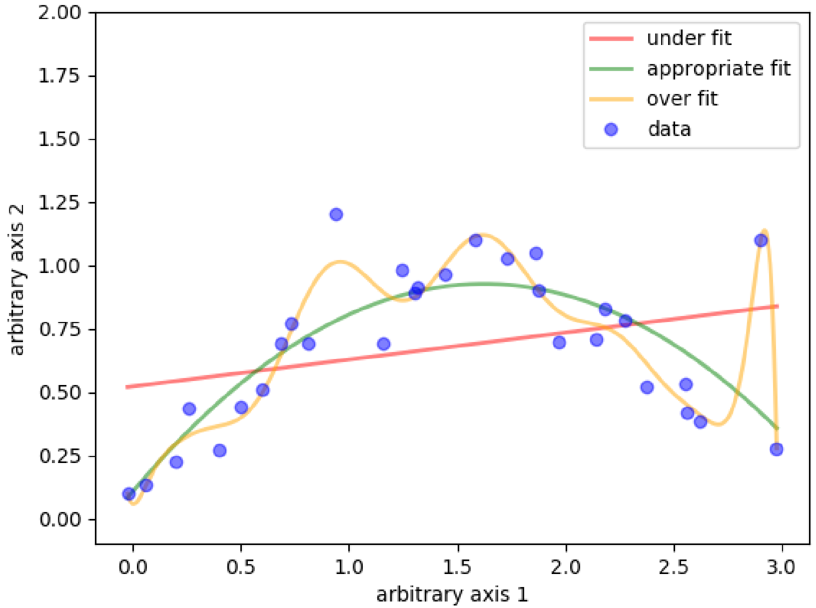
Possibly a too complex model (=overfitting)?



Possibly an appropriate model?



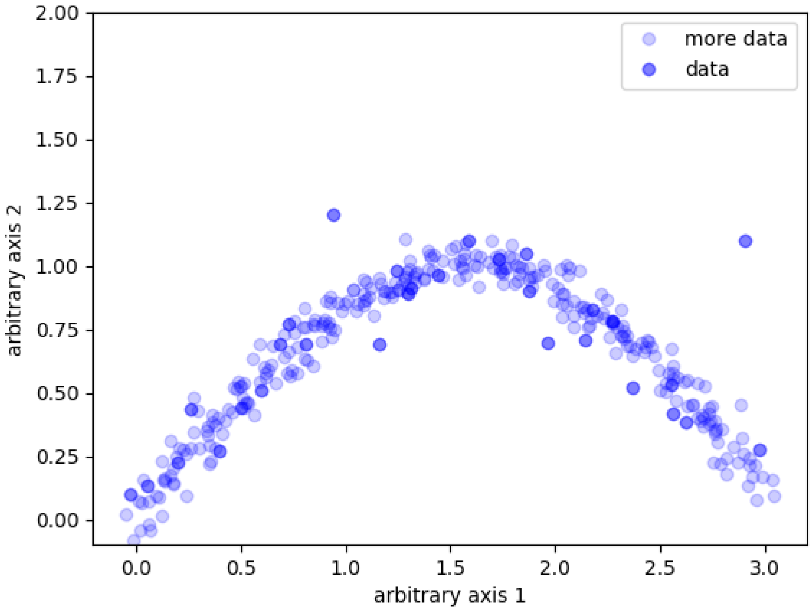
All models for a direct comparison …



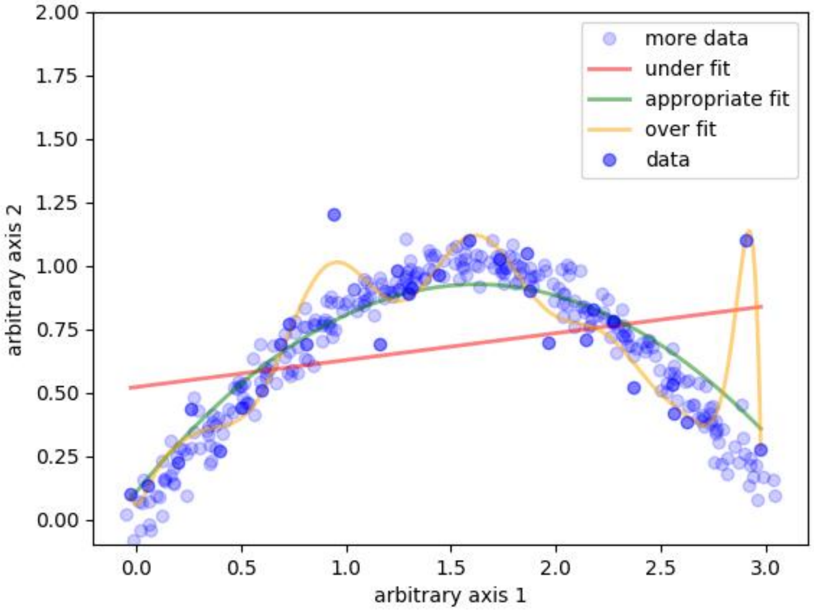
FH University of Applied Sciences TECHNIKUM WIEN

… but why is the complex (orange) model not necessarily the best?
It is closest to the data points!?!

Imagine that suddenly we have more data (from the same source) …



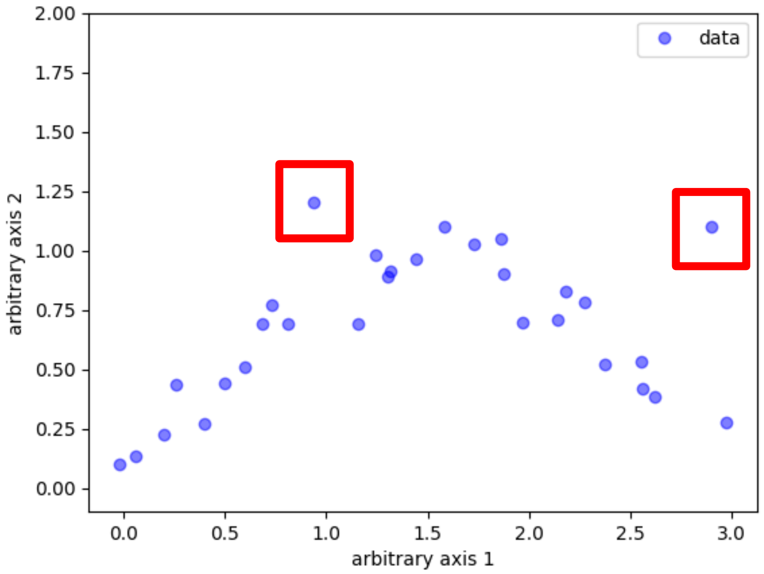
Now the orange model seems not such a great choice anymore …

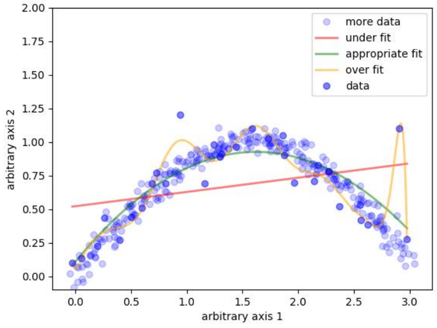


FH University of Applied Sciences TECHNIKUM WIEN

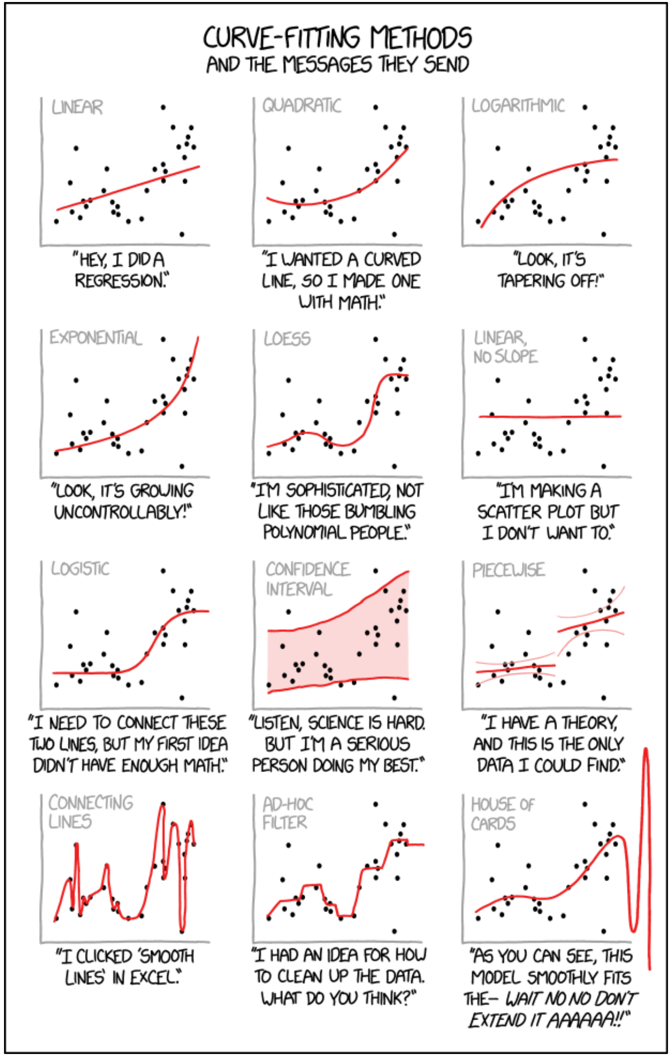
The orange model tries to capture outliers …

It is overly complex. It has “over fitted“ the training data and fails on the test data.





FH University of Applied Sciences TECHNIKUM WIEN



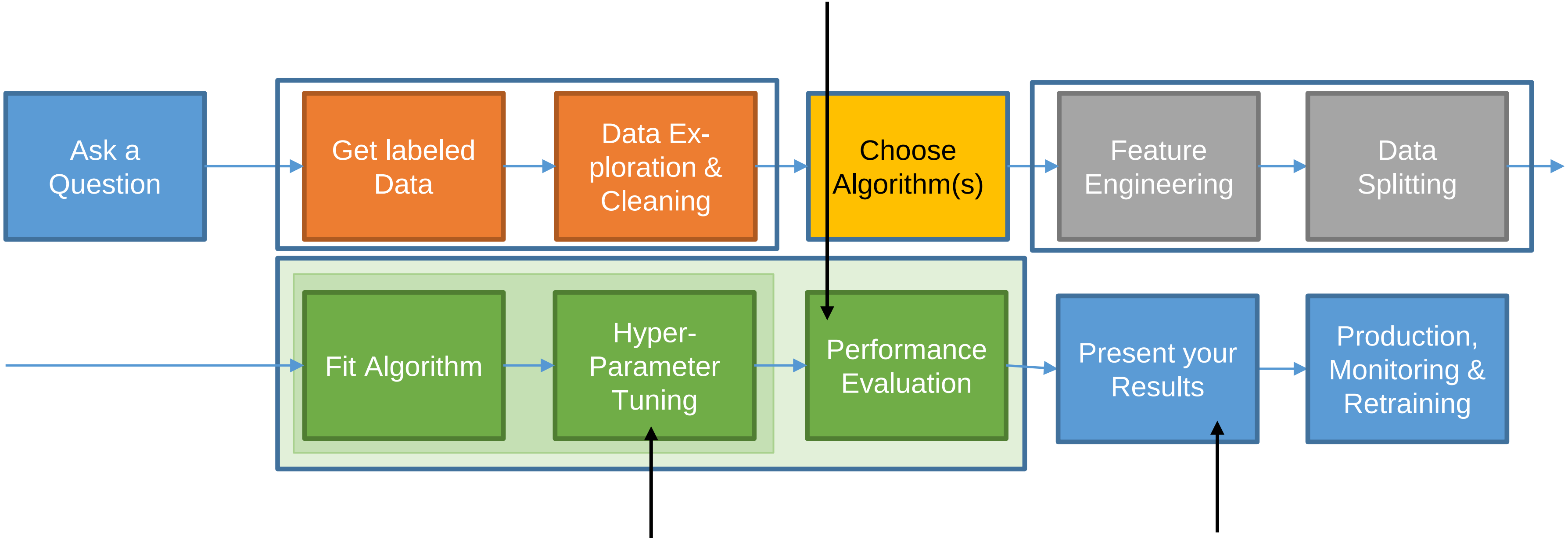
FH University of Applied Sciences TECHNIKUM WIEN

Contents

FH University of Applied Sciences TECHNIKUM WIEN

Recap

Holdout Validation
LpoCV
Cross-Validation
Monte Carlo Cross Validation



Holdout-Analog
Holdout/Cross-Validation Combination
Nested Cross-Validation
HPs: k, Metric, Weights

Decision Surfaces
Roc-Curves

References

- Géron A. (2017): Hands-On Machine Learning with Scikit-Learn & Tensorflow. – O‘Reilly.
- James G., Witten D., Hastie T., Tibshirani R. (2017): An introduction to Statistical Learning. – Springer.
- Kuhn M., Johnson K. (2016): Apllied Predictive Modeling. – Springer.
- Meyer D., Plank P.: Slides Master Data Science @ FHTW

FH University of Applied Sciences TECHNIKUM WIEN